UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/718,867	11/21/2003	Dong Ho Song	351306-991110	1481

26379 7590 05/12/2010
DLA PIPER LLP (US)
2000 UNIVERSITY AVENUE
EAST PALO ALTO, CA 94303-2248

EXAMINER
WANG, BEN C

ART UNIT	PAPER NUMBER
2192	

NOTIFICATION DATE	DELIVERY MODE
05/12/2010	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

PatentDocketingUS-PaloAlto@dlapiper.com

Office Action Summary	**Application No.** 10/718,867	**Applicant(s)** SONG ET AL.	
	Examiner BEN C. WANG	**Art Unit** 2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *27 January 2010*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-46* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-46* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application

6) ☐ Other: _____.

DETAILED ACTION

1. Applicant's amendment dated January 27, 2010, responding to the Office action mailed July 27, 2009 provided in the rejection of claims 1-46, wherein claims 1, 2, and 8 have been amended to overcome 35 USC § 101 rejections and claim objections.

Claims 1-46 remain pending in the application and which have been fully considered by the examiner.

Applicant's arguments with respect to claims currently amended have been fully considered but are not persuasive. Please see the section of "Response to Arguments" for details.

Response to Arguments

2. Applicant's arguments filed on January 27, 2010 have been fully considered but they are not persuasive.

In the remarks, Applicant argues that, for examples:

(A.1) Applicant contends that the prior arts cited in either *Blaser* or *Schaefer* do not expressly or inherently disclose "*a privatized virtual file resource created from an operating system file system wherein the application software only accesses the privatized virtual file resource*" and "*a privatized virtual registry created from an operating system registry system wherein the application software only* *access the privatized virtual registry*" (Remarks, on page 17-18 – emphasis added)

Examiner Responses:

Examiner respectfully disagrees. Firstly, Examiner mainly depends on *Schaefer* reference to disclose *the privatized functionalities* (please see Exhibit **A** below with Operating System Guard 100 and Private Contexts and further in the paragraph [0019] - each application (*interpreted as the application software*) is able to run in a **private context** (*interpreted as only access its private contexts*) within the system and two separate applications 52, 54, or two instances of the same application, can be provided **private contexts** in which they will appear to have separate or differing copies of system services, configuration and data – emphasis added) via the Operating System Guard 100 (see Fig. 1, OSGurad 100; [0027] – if an attempt is made to modify the value, the Operating System Guard will allow the modification to occur to itself only) as stated in the previous Office Action dated July 27, 2009.

Secondly, *Schaefer* indeed discloses a privatized virtual file resource created from an operating system file system wherein the application software only accesses the privatized virtual file resource (e.g., [0040] - … a virtual file system (*interpreted as the privatized virtual file resource*) much like the virtual registry … Whenever the application accesses or modifies any files, the Operation System Guard checks if the file must be redirected, and if so, in the preferred embodiment redirects the request to a location specified in the Operating System Guard configuration).

Thirdly, *Schaefer* further teaches a privatized virtual registry created from an operating system registry system wherein the application software only access the privatized virtual registry (e.g., [0027] - ... a virtual Windows® Registry (*interpreted as the privatized virtual registry*) component, which will provide a full function registry to an application (*the application software*), but prevent modification to the underlying system registry. All keys that an application expects to access will be present, but may only exist in the virtual registry ...).

Hence, *Schaefer* indeed discloses the claim limitation recited above.

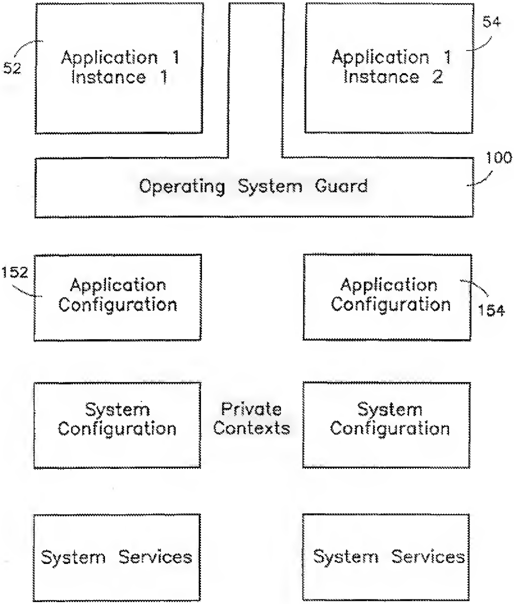

Fig. 2

Exhibit A – excerpted from Figure 2 in *Schaefer* reference

3. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Claim Rejections – 35 USC § 103(a)

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made

4. Claims 1-7, 9-30, and 32-46 are rejected under 35 U.S.C. 103(a) as being unpatentable over Blaser et al. (Pat. No. US 7,117,495 B2) (hereinafter 'Blaser') in view of Stuart Schaefer (Pub. No. US 2002/0174215 A1) (hereinafter 'Schaefer')

5. **As to claim 1** (Currently Amended), Blaser discloses a system for executing application software (e.g., Fig. 1, element 104 - Applications; Fig. 2, element 200 – Application) on a operating system within a secured run-time environment (e.g., Col. 11, Lines 20-38 – Secure Applications; Col. 11, Line 63 through Col. 12, Line 7 – Secure Base OS; Col. 13, Lines 1-4, 16-23) without affecting an application software resources of a client computer (e.g., Col. 7, Lines 24-34; Col. 13, Lines 30-33), the system comprising:

- a client computer (e.g., Col. 2, Lines 50-52 – computer systems having facilities for providing virtual portions of files systems and configuration settings to application – emphasis added);
- an application wrapper executing on the client computer (e.g., Fig. 1, element 106 – Layering System Libraries/Software; Fig. 2, elements 202 – Layer B, 204 – Layer A; Fig. 3, elements 300 – FSL (File System Layer) Management Application, 306 – FSL API Library, 310 – FSL Compression Library, 312 – FSL System Driver), wherein the application wrapper shields the application software resources (e.g., Col. 4, Lines 15-26 – layering system libraries and/or software intercepts files system and registry access from applications), whereby the secured run-time environment for executing the application software is created and the application software resources are protected (e.g., Col. 11, Lines 20-38 – Secure Applications; Col. 11, Line 63 through Col. 12, Line 7 – Secure Base OS; Col. 13, Lines 1-4, 16-23); and

- the application wrapper further comprising a privatized virtual file resource created from an operating system file system (e.g., Fig. 3; Col. 13, Line 58 through Col. 14, Line 11 – an FSL system driver is installed "on top" of the operating system in order to have first processing priority for file system accesses), a privatized virtual registry created from an operating system registry system (e.g., Fig. 3; Col. 13, Line 58 through Col. 14, Line 11 – an FSL system driver is installed "on top" of the operating system in order to have first processing priority for registry), a privatized operating system shared component resource (e.g., Col. 5, Lines 18-32; Col. 6, Line 62 through Col. 7, Line 17), a privatized application configuration resource (e.g., Col. 2, Lines 50-52 – having facilities for providing configuration settings to applications; Col. 3, Lines 42-46, 51-58; Col. 4, Lines 23-26 – a layer manager application may be provided to permit control and configuration of the layering system software through a management API and library; Col. 5, Lines 23-29; Col. 7, Lines 48-53; Col. 9, Lines 6-11; Col. 11, Lines 16-19) and a privatized environmental resources for application variables (e.g., Col. 8, Lines 48-61 – for example, Windows operating systems set the "WINDIR" environment variable to the location of the OS system subtree…; Col. 8, Line 67 through Col. 9, Line 11)

Although Blaser discloses that applications can be protected from unauthorized access through the use of a layer system (e.g., Col. 11, Lines 20-22) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Operating System Abstraction and Protection Layer*, Schaefer discloses the followings:

- wherein the application software only access the privatized virtual file resource (e.g., Fig. 2, elements 52 – Application Instance 1; 100 – Operating System Guard; 152 – Application Configuration, Private Contexts; Private Contexts; Fig. 4, element of 'Virtual File System Manager'; 'Virtual Environment Manager'; [0040] - … a virtual file system much like the virtual registry … Whenever the application accesses or modifies any files, the Operation System Guard checks if the file must be redirected, and if so, in the preferred embodiment redirects the request to a location specified in the Operating System Guard configuration; [0041] – If an application tries to create a new file or open an existing file for writing on a user's local drive, the Operating System Guard must ensure that the file is actually created or modified in the redirected location …; [0019] - … each application is ab le to run in a private context within the system. To the application, it has its own private view of what the system looks like and its behavior); and
- wherein the application software only accesses the privatized virtual registry (e.g., Fig. 2, elements 52 – Application Instance 1; 100 – Operating System Guard; 152 – Application Configuration, Private Contexts; Private Contexts; [0027] - … a virtual Windows® Registry (*the privatized virtual registry*) component, which will provide a full function registry to an application (*the application software*), but

> prevent modification to the underlying system registry. All keys that an application expects to access will be present, but may only exist in the virtual registry ...; [0019] - ... each application is ab le to run in a private context within the system. To the application, it has its own private view of what the system looks like and its behavior;
>
> **NOTE:** please see Exhibit **A** above with Operating System Guard 100 and Private Contexts and further in the paragraph [0019] - each application (*interpreted as the application software*) is able to run in a **private context** (*interpreted as only access its private contexts*) within the system and two separate applications 52, 54, or two instances of the same application, can be provided **private contexts** in which they will appear to have separate or differing copies of system services, configuration and data; also see Fig. 1, OSGurad 100; [0027] – if an attempt is made to modify the value, the Operating System Guard will allow the modification to occur to itself only – emphasis added)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Schaefer into the Blaser's system to further provide other limitations stated above in Blaser system.

The motivation is that it would further enhance the Blaser's system by taking, advancing and/or incorporating Schaefer's system which offers significant advantages that the operating system abstraction and protection layer presents an environment to an application that appears to be an installation environment without performing an installation, whereby a “pseudo installations” is created in which all of the settings are brought into a virtual environment at the time the application runs as once suggested by Schaefer (e.g., [0007])

6. **As to claim 2** (Currently Amended) (incorporating the rejection in claim 1), Blaser discloses privatized virtual file resource (e.g., Col. 16, Line 11 through Col. 18, Line 12 – File System Calls) further comprising:

- intercepting file input/output (I/O) request generated by one or more processes (e.g., Fig. 5, step 502 – “wait for read request”; Fig. 6, step 602 – “wait for write request”);
- establishing a process identifier (ID) for the intercepted file I/O request (e.g., Col. 9, Lines 34-38, 58-63; Col. 30, Lines 52-53, 58-61);
- comparing process ID to establish operating system process and secured run-time process (e.g., Fig. 5, step 504 – “is file reference maintained in an enabled layer?”; Fig. 6, step 604 – “is file reference to be captured to an enabled layer?”; Col. 6, Lines 37-39; Col. 15, Lines 16-26);
- establishing a process ID as operating system process and secured run-time process (e.g., Col. 9, Lines 34-38, 58-63; Col. 30, Lines 52-53, 58-61);
- servicing the file I/O request for all process ID established as secured run-time process (e.g., Fig. 5, steps 508, 510, 512; Fig. 6, steps 608, 610, 612); redirecting the file I/O request to operating system service for process ID established as operating system process (e.g., Fig. 5, step 506

– "use reference of request to execute regular read"; Fig. 6, step 606 – "use reference of request to execute regular write"; Col. 4, Lines 15-26; Col. 9, Lines 29-33; Col. 14, Line 64 through Col. 15, Line 3; Col. 15, Line 66 through Col. 16, Line 3; Col. 16, Lines 41-44; Col. 17, Lines 5-13, 24-28);

- rejecting the file I/O request on secured run-time process resources for process ID established as operating system process (e.g., Col. 4, Lines 15-26; Col. 9, Lines 29-33; Col. 14, Line 64 through Col. 15, Line 3; Col. 15, Line 66 through Col. 16, Line 3; Col. 16, Lines 41-44; Col. 17, Lines 5-13, 24-28);
- comparing process ID established as secured run-time process to further establish process resources corresponding to process ID (e.g., Col. 15, Lines 16-26);
- establishing corresponding process resources within secured run-time resources (e.g., Fig. 5, steps 508, 510, 512; Fig. 6, steps 608, 610, 612); and
- rejecting the file I/O request on secured run-time process resources for process ID established as secured run-time process and process ID belongs to other process resources (e.g., Col. 4, Lines 15-26; Col. 9, Lines 29-33; Col. 14, Line 64 through Col. 15, Line 3; Col. 15, Line 66 through Col. 16, Line 3; Col. 16, Lines 41-44; Col. 17, Lines 5-13, 24-28).

7. **As to claim 3** (Original) (incorporating the rejection in claim 1), Blaser discloses privatized virtual registry (e.g., Col. 18 Line 13 through Col. 20, Line 18 – Registry Calls) further comprises:

- privatizing virtual registry system by intercepting registry I/O request generated by several process (Fig. 7, step 702 – "waiting for request for registry setting"; Col. 6, Lines 33-37);
- establishing process ID for the intercepted registry I/O request (e.g., Col. 9, Lines 34-38, 58-63; Col. 30, Lines 52-53, 58-61);
- comparing process ID to establish operating system process and secured run-time process (e.g., Fig. 7, step 704 – "is registry request to be captured to an enabled layer?"; Col. 6, Lines 37-39; Col. 15, Lines 16-26);
- establishing process ID as operating system process and secured run-time process (e.g., Fig. 7, step 708 – identify destination layer; Col. 6, Lines 41-43);
- servicing the registry I/O request for all process ID established as secured run-time process (e.g., Fig. 7, elements 712 – create virtual registry setting in layer, 716- create delete entry in virtual registry, or delete entry if isolated to layer, 720 – set virtual registry setting in layer; Col. 6, Lines 44-55);
- redirecting the registry I/O request to operating system service for process ID established as operating system process (e.g., Fig. 7, element 706 – execute normal registry function; Col. 6, Lines 39-41; Col. 14, Line 64

through Col. 15, Line 3; Col. 15, Lines 16-26; Col. 15, Line through Col. 16, Line 11 – the structures and hooks);

- rejecting the registry I/O request on secured run-time process resources for process ID established as operating system process (e.g., Fig. 7, element 706 – execute normal registry function; Col. 6, Lines 39-41; Col. 15, Lines 16-26);
- comparing process ID established as secured run-time process to further establish process resources corresponding to process ID (e.g., Col. 15, Lines 16-26);
- establishing corresponding process resources within secured run-time resources (e.g., Fig. 7, elements 712, 716, 720); and
- rejecting the registry I/O request on secured run-time process resources for process ID established as secured run-time process and process ID belongs to other process resources (e.g., Fig. 7, elements 706 – execute normal registry function; Col. 6, Lines 39-41, 708 – identify destination layer; Col. 6, Lines 41-43)

8. **As to claim 4** (Currently Amended) (incorporating the rejection in claim 1), Blaser discloses privatizing operating system shared component resource (e.g., Col. 5, Lines 18-32; Col. 6, Line 62 through Col. 7, Line 17) further comprising:

- searching secured application process for injecting component hooker (e.g., Col. 9, Lines 43-46; Col. 15, Line 46 through Col. 16, Line 11);

- checking the secured application process to establish whether the process is injected with component hooker (e.g., Col. 16, Lines 36-38, 45-47);
- establishing the secured application process as new secured application process for the secured application process not injected with component hooker (e.g., Col. 8, Lines 11-18; Col. 19, Lines 28-36);
- injecting component hooker to new secured application process to intercept component process (e.g., Fig. 3, element 312 – FSL System Driver; Col. 9, Lines 6-11; Col. 13, Line 62 through Col. 14, Line 2; Col. 15, Lines 42-44 – all of the file system entry points are hooked in; the drives to be redirected are hooked in; all of the Registry entry points are hooked in);
- repeating component hooker injection for all the new secured application process (e.g., Col. 12, Lines 29-31)

9. **As to claim 5** (Currently Amended) (incorporating the rejection in claim 1), Blaser discloses privatizing operating system shared component resource further comprising:

- initializing component redirection table to provide component redirecting information (e.g., Fig. 7, element 712 – create virtual registry setting in layer; Col. 14, Lines 17-22, 30-31 – *FileRedirect*, 36-37 – *RegRedirect*; Col. 20, lower portion for Function/Description List – *FSLCreate()* – create the layer redirection area in the file system);

- registering virtual component required for the secured application (e.g., Fig. 7, element 716 – set virtual registry setting in layer; Col. 27 – Function/Description List – *FSLSetLayerInfo()* – if *fileRedirect* is specified, set the value of the proper registry value… create the redirect root keys..);
- adding redirecting information to the component redirection table for the execution of each selected the secured run-time application (e.g., Col. 25 – Function/Description list – *FSLRegCreateKeyEx()* – create a registry path to the layer's redirection area using the layer's redirect path, its name, …. Create the key in the redirection area);
- removing component redirecting information from the component redirection table for the termination of each the secured run-time application (e.g., Fig. 7, element 716 – delete entry if isolated to layer)

10. **As to claim 6** (Original) (incorporating the rejection in claim 1), Blaser discloses privatizing operating system shared component resource further comprising:

- intercepting component process function for replacing component search path with secured application resource path (e.g., Fig. 5, element 502; Fig. 6, element 602; Fig. 7, element 702);
- replacing component search path with private resource path to load the component from the secured application resource path (e.g., Col. 5, Lines 60-64; Col. 6, Lines 12-17; Col. 8, Lines 48-52); and

- creating new process for the intercepted component with the replaced secured application resource path (e.g., Col. 15, Lines 58-60; Col. 16, Lines 19-48; Col. 16, Line 53 through Col. 17, Line 36; Col. 18, Lines 26-61)

11. **As to claim 7** (Original) (incorporating the rejection in claim 1), Blaser discloses privatizing operating system shared component resource further comprising:

- intercepting component call for replacing component registry path with the privatized virtual registry path (e.g., Fig. 7, element 702);
- searching component redirection table for the component redirecting information (e.g., Col. 18, Lines 26-28; Col. 19, Lines 24-26; Col. 19, Line 37 through Col. 20, Line 3);
- replacing component registry path with the privatized virtual registry path retrieved from the component redirection table (e.g., Col. 15, Lines 58-60; Col. 16, Lines 19-48; Col. 16, Line 53 through Col. 17, Line 36; Col. 18, Lines 26-61);
- returning the intercepted call to the requested call with the replaced secured application registry path for addressing the component location from the privatized virtual registry system and further the component is addressed to load from the privatized virtual file system (e.g., Fig. 7, element 708; Col. 6, Lines 41-43);

- redirecting the component call as it is to the requested call for the component call originated from non secured run-time application and for the component call, which do not have redirecting information in the component redirection table (e.g., Fig. 7, element 706 – "execute normal registry function"; Col. 6, Lines 39-41)

12. **As to claim 9** (Original) (incorporating the rejection in claim 1), Blaser discloses privatized application configuration resource further comprises:

- monitoring file I/O request for configuration file to provide separate configuration file (e.g., Fig. 5, element 502; Fig. 6, element 602; Col. 9, Lines 6-11);
- searching and retrieving configuration file from secured application resources (e.g., Col. 2, Lines 50-52 – having facilities for providing configuration settings to applications; Col. 3, Lines 42-46, 51-58; Col. 4, Lines 23-26 – a layer manager application may be provided to permit control and configuration of the layering system software through a management API and library); and
- serving application configuration file to requesting process (e.g., Col. 5, Lines 23-29; Col. 7, Lines 48-53; Col. 11, Lines 16-19)

13. **As to claim 10** (Original) (incorporating the rejection in claim 1), Blaser discloses privatized environmental resources (e.g., Col. 8, Lines 48-61 – for example, Windows operating systems set the "WINDIR" environment variable to

the location of the OS system subtree…; Col. 8, Line 67 through Col. 9, Line 11) further comprises:

- intercepting environment variable request to supply private values to secured application process (e.g., Fig. 5, element 502; Fig. 6, element 602; Col. 9, Lines 6-11);
- verifying process ID to establish the process ID as operating system process or secured application process (e.g., Fig. 5, step 504 – "is file reference maintained in an enabled layer?"; Fig. 6, step 604 – "is file reference to be captured to an enabled layer?"; Col. 6, Lines 37-39; Col. 15, Lines 16-26);
- redirecting the call for process ID established as operating system process (e.g., Fig. 5, element 506; Fig. 6, element 606);
- reading variable data from secured application resource and returning the value to requested process for read variable calls requested by the secured application process (e.g., Fig. 5, element 512; Col. 5, Lines 60-64) ;
- searching the requesting write variable in secured application resource to find the presence of requesting write variable (e.g., Col. 23 – Function/Description List – *FSLFindCloseVariable(), FSLFindFirstVariable(), FSLFindNextVariable(), FSLGetVariable()*);
- creating variable with variable data within secured application resource and returning the status to requested process for variable do not exist in

secured application resource (e.g., Col. 20, lower portion – Function/Description List – *FSLAddVariable()*); and

- updating variable with variable data within secured application resource and returning the status to requested process for variable exist in secured application resource (e.g., Col. 8, Line 61 through Col. 9, Line 4)

14. **As to claim 11** (Original) (incorporating the rejection in claim 1), Blaser discloses the application wrapper further comprises selectively allowing the application software to interact operating system resources directly during the application software executing under the secured run-time environment (e.g., Fig. 3, element 304 – Other Applications; Col. 14, Lines 6-11)

15. **As to claim 12** (Original) (incorporating the rejection in claim 1), Blaser discloses the application Wrapper further comprises selectively allowing the application software to interact with other application software resources directly during the application software executing under the secured run-time environment (e.g., Fig. 3, element 304 – Other Applications; Col. 14, Lines 6-11)

16. **As to claim 13** (Original) (incorporating the rejection in claim 1), Blaser discloses the application wrapper further comprises providing a run-time environment to the application software that is visible to an operating system run-time environment without having the application software run-time resources, whereby the application software resources is simulated to the secured run-time

environment during the execution of the application software (e.g., Col. 12, Lines 8-20 – all further changes are recorded to a layer and not to the underlying file systems and OS resources)

17. **As to claim 14** (Original) (incorporating the rejection in claim 13), Blaser discloses the system and method further comprising means (e.g., Fig. 1, element 106 – Layering System Libraries/Software; Fig. 2, elements 202 – Layer B, 204 – Layer A; Fig. 3, elements 300 – FSL (File System Layer) Management Application, 306 – FSL API Library, 310 – FSL Compression Library, 312 – FSL System Driver) for protecting the behavior of the application software from other application and the operating system (e.g., Col. 11, Lines 20-38 – Secure Applications; Col. 11, Line 63 through Col. 12, Line 7 – Secure Base OS; Col. 13, Lines 1-4, 16-23)

18. **As to claim 15** (Original) (incorporating the rejection in claim 13), Blaser discloses the system and method further comprising means (e.g., Fig. 1, element 106 – Layering System Libraries/Software; Fig. 2, elements 202 – Layer B, 204 – Layer A; Fig. 3, elements 300 – FSL (File System Layer) Management Application, 306 – FSL API Library, 310 – FSL Compression Library, 312 – FSL System Driver) for eliminating the application conflicts from other running application software (e.g., Col. 2, Lines 42-46; Col. 3, Lines 51-58; Col. 8, Lines 11-18)

19. **As to claim 16** (Original) (incorporating the rejection in claim 13), Blaser discloses the system and the method further comprising means for executing multiple instance of the single application software (e.g., Col. 8, Lines 6-11)

20. **As to claim 17** (Original) (incorporating the rejection in claim 1), Blaser discloses the system and the method wherein the application wrapper further comprising maintaining the application software resources away from the operating system resources (e.g., Col. 4, Lines 15-26 – layering system libraries and/or software intercepts files system and registry access from applications), whereby the operating system resources is protected from the application software resources (e.g., Col. 11, Lines 20-38 – Secure Applications; Col. 11, Line 63 through Col. 12, Line 7 – Secure Base OS; Col. 13, Lines 1-4, 16-23)

21. **As to claim 18** (Original) (incorporating the rejection in claim 1), Blaser discloses the system and the method wherein the application wrapper further comprises permitting full access to the application software that requires to access for variation occurs to the application software resources within the application wrapper (e.g., Col. 4, Lines 15-26; Col. 8, Lines 19-23, 48-55)

22. **As to claim 19** (Original) (incorporating the rejection in claim 18), Blaser discloses the system and the method further comprising means for keeping the state of secured run-time environment to the application software (e.g., Col. 6,

Lines 3-10; Col. 7, Lines 9-12; Col. 10, Lines 2-24; Col. 11, Line 63 through Col. 12, Line 7)

23. **As to claim 20** (Original) (incorporating the rejection in claim 18), Blaser discloses the system and the method further comprising means for updating the application software resources required by the application software (e.g., Col. 5, Lines 18-32 - shared libraries, system accessible configuration, and version control are managed by the layering subsystem)

24. **As to claim 21** (Original) (incorporating the rejection in claim 1), Blaser discloses the system and the method wherein the application wrapper monitors the application run-time request to determine the required the application software resources for execution (e.g., Fig. 1, element 106 – Layering System Libraries/Software; Col. 4, Lines 15-26)

25. **As to claim 22** (Original) (incorporating the rejection in claim 21), Blaser discloses the system and the method further comprising means for receiving the application software resources to execute the application software in the secured run-time environment (e.g., Fig. 1, element 106 – Layering System Libraries/Software; Fig. 2, elements 202 – Layer B, 204 – Layer A; Fig. 3, elements 300 – FSL (File System Layer) Management Application, 306 – FSL API Library, 310 – FSL Compression Library, 312 – FSL System Driver)

26. **As to claim 23** (Original) (incorporating the rejection in claim 21), Blaser discloses the system and the method further comprising means for incrementally executing the application software in the secured run-time environment (e.g., Col. 3, Line 59 through Col. 4, Line 6; Col. 5, Lines 18-32 – shared libraries, system accessible configuration, and version control are managed by the layering subsystem)

27. **As to claim 24** (Previously Presented), Blaser discloses a method for executing application software on a operating system within a secured run-time environment (e.g., Col. 11, Lines 20-38 – Secure Applications; Col. 11, Line 63 through Col. 12, Line 7 – Secure Base OS; Col. 13, Lines 1-4, 16-23) without affecting an application software resources of a client computer (e.g., Col. 7, Lines 24-34; Col. 13, Lines 30-33), the client computer comprising an application wrapper (Fig. 1, element 106 – Layering System Libraries/Software; Fig. 2, elements 202 – Layer B, 204 – Layer A; Fig. 3, elements 300 – FSL (File System Layer) Management Application, 306 – FSL API Library, 310 – FSL Compression Library, 312 – FSL System Driver), wherein the application wrapper shields the application software resources (e.g., Col. 4, Lines 15-26 – layering system libraries and/or software intercepts files system and registry access from applications), whereby a the secured run-time environment for executing an the application software is created and the application software resources is protected (e.g., Col. 11, Lines 20-38 – Secure Applications; Col. 11, Line 63

through Col. 12, Line 7 – Secure Base OS; Col. 13, Lines 1-4, 16-23), the method further comprising:

- privatizing virtual file resource created from an operating system file system (e.g., Fig. 3; Col. 13, Line 58 through Col. 14, Line 11 – an FSL system driver is installed "on top" of the operating system in order to have first processing priority for file system accesses);
- privatizing virtual registry created from an operating system registry system (e.g., Fig. 3; Col. 13, Line 58 through Col. 14, Line 11 – an FSL system driver is installed "on top" of the operating system in order to have first processing priority for registry);
    - privatizing operating system shared component resource (e.g., Col. 5, Lines 18-32; Col. 6, Line 62 through Col. 7, Line 17);
    - privatizing application configuration resource (e.g., Col. 2, Lines 50-52 – having facilities for providing configuration settings to applications; Col. 3, Lines 42-46, 51-58; Col. 4, Lines 23-26 – a layer manager application may be provided to permit control and configuration of the layering system software through a management API and library; Col. 5, Lines 23-29; Col. 7, Lines 48-53; Col. 9, Lines 6-11; Col. 11, Lines 16-19); and
    - privatizing environmental resources for application variables (e.g., Col. 8, Lines 48-61 – for example, Windows operating systems set the "WINDIR" environment variable to the location of the OS system subtree…; Col. 8, Line 67 through Col. 9, Line 11)

Further, Blaser discloses that applications can be protected from unauthorized access through the use of a layer system (e.g., Col. 11, Lines 20-22) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Operating System Abstraction and Protection Layer* Schaefer discloses the followings:

- wherein the application software only access the privatized virtual file resource (e.g., Fig. 2, elements 52 – Application Instance 1; 100 – Operating System Guard; 152 – Application Configuration, Private Contexts; Private Contexts; Fig. 4, element of 'Virtual File System Manager'; 'Virtual Environment Manager'; [0040] - … a virtual file system much like the virtual registry … Whenever the application accesses or modifies any files, the Operation System Guard checks if the file must be redirected, and if so, in the preferred embodiment redirects the request to a location specified in the Operating System Guard configuration; [0041] – If an application tries to create a new file or open an existing file for writing on a user's local drive, the Operating System Guard must ensure that the file is actually created or modified in the redirected location …; [0019] - … each application is ab le to run in a private context within the system. To the application, it has its own private view of what the system looks like and its behavior); and
- wherein the application software only accesses the privatized virtual registry (e.g., Fig. 2, elements 52 – Application Instance 1; 100 – Operating System Guard; 152 – Application Configuration, Private

Contexts; Private Contexts; [0027] - … a virtual Windows® Registry (*the privatized virtual registry*) component, which will provide a full function registry to an application (*the application software*), but prevent modification to the underlying system registry. All keys that an application expects to access will be present, but may only exist in the virtual registry …; [0019] - … each application is ab le to run in a private context within the system. To the application, it has its own private view of what the system looks like and its behavior;

NOTE: please see Exhibit **A** above with Operating System Guard 100 and Private Contexts and further in the paragraph [0019] - each application (*interpreted as the application software*) is able to run in a **private context** (*interpreted as only access its private contexts*) within the system and two separate applications 52, 54, or two instances of the same application, can be provided **private contexts** in which they will appear to have separate or differing copies of system services, configuration and data; also see Fig. 1, OSGurad 100; [0027] – if an attempt is made to modify the value, the Operating System Guard will allow the modification to occur to itself only – emphasis added)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Schaefer into the Blaser's system to further provide other limitations stated above in the Blaser system:

The motivation is that it would further enhance the Blaser's system by taking, advancing and/or incorporating Schaefer's system which offers significant advantages that the operating system abstraction and protection layer presents an environment to an application that appears to be an installation environment without performing an installation, whereby a "pseudo installations" is created in which all of the settings are brought into a virtual environment at the time the application runs as once suggested by Schaefer (e.g., [0007])

28. **As to claim 25** (Original) (incorporating the rejection in claim 24), please refer to claim **2** as set forth accordingly.

29. **As to claim 26** (Original) (incorporating the rejection in claim 24), please refer to claim **3** as set forth accordingly.

30. **As to claim 27** (Original) (incorporating the rejection in claim 24), please refer to claim **6** as set forth accordingly.

31. **As to claim 28** (Previously Presented) (incorporating the rejection in claim 24), please refer to claim **4** as set forth accordingly.

32. **As to claim 29** (Previously Presented) (incorporating the rejection in claim 24), please refer to claim **5** as set forth accordingly.

33. **As to claim 30** (Original) (incorporating the rejection in claim 24), please refer to claim **7** as set forth accordingly.

34. **As to claim 32** (Original) (incorporating the rejection in claim 24), please refer to claim **9** as set forth accordingly.

35. **As to claim 33** (Original) (incorporating the rejection in claim 24), please refer to claim **10** as set forth accordingly.

36. **As to claim 34** (Original) (incorporating the rejection in claim 24), please refer to claim **11** as set forth accordingly.

37. **As to claim 35** (Original) (incorporating the rejection in claim 24), please refer to claim **12** as set forth accordingly.

38. **As to claim 36** (Original) (incorporating the rejection in claim 24), please refer to claim **13** as set forth accordingly.

39. **As to claim 37** (Original) (incorporating the rejection in claim 36), please refer to claim **14** as set forth accordingly.

40. **As to claim 38** (Original) (incorporating the rejection in claim 36), please refer to claim **15** as set forth accordingly.

41. **As to claim 39** (Original) (incorporating the rejection in claim 36), please refer to claim **16** as set forth accordingly.

42. **As to claim 40** (Original) (incorporating the rejection in claim 24), please refer to claim **17** as set forth accordingly.

43. **As to claim 41** (Original) (incorporating the rejection in claim 24), please refer to claim **18** as set forth accordingly.

44. **As to claim 42** (Original) (incorporating the rejection in claim 41), please refer to claim **19** as set forth accordingly.

45. **As to claim 43** (Original) (incorporating the rejection in claim 41), please refer to claim **20** as set forth accordingly.

46. **As to claim 44** (Original) (incorporating the rejection in claim 24), please refer to claim **21** as set forth accordingly.

47. **As to claim 45** (Original) (incorporating the rejection in claim 44), please refer to claim **22** as set forth accordingly.

48. **As to claim 46** (Original) (incorporating the rejection in claim 44), please refer to claim **23** as set forth accordingly.

49. Claims 8 and 31 are rejected under 35 U.S.C. 103(a) as being unpatentable over Blaser in view of Schaefer and further in view of Osman et al., (*The Design and Implementation of Zap: A System for Migrating Computing Environments, 2002, ACM, pp. 361-376*) (hereinafter 'Osman') and Galen C. Hunt (Pub. No. US 2002/0072830 A1) (hereinafter 'Hunt')

50. **As to claim 8** (Currently Amended) (incorporating the rejection in claim 1), although Blaser discloses hooking/redirected structures (e.g., Col. 15, Line 38 through Col. 20, Line 18) and its intercept/redirected mechanism for file system and registry accesses from applications (e.g., Col. 4, Lines 15-26), but Blaser and Schaefer do not explicitly disclose the limitations stated below.

However, in an analogous art of *a system for migrating computing environments*, Osman discloses intercepting a remote procedure call (RPC) message call for replacing component information with privatized virtual component information (e.g., Table 1 – Pod principles applied to resources – Network Resource; Sec. 4 – Zap Architecture, 2nd Para., Lines 1-12; Sec. 4.5 – Network Virtualization and Migration, 1st Para., Lines 4-11; 8th Para., Lines 1-8 – connection virtualization works by intercepting system calls for connection setup requests from the application to the transport protocol and replacing relevant physical addresses with virtual address); searching component redirecting information from the component redirection table; replacing RPC message with the privatized virtual component information retrieved from the component redirection table returning the intercepted RPC message call to the requested call with the replaced message (e.g., Sec. 4.5 – Network Virtualization and Migration, 9th Para., Lines 1-9 – Connection translation works by intercepting packets below the transport protocol layer and translating the virtual address in the packet headers to physical address).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Osman into the Blaser-Schaefer's system to further provide the limitations stated above in the Blaser-Schaefer's system.

The motivation is that it would enhance the Blaser-Schaefer's system by taking, advancing and/or incorporating Osman's system which provides a framework for network virtualization as once suggested by Osman (e.g., Sec. 4.5 – Network Virtualization and Migration, 1st Para.).

Although Osman discloses network virtualization (e.g., Sec. 4.5) but Blaser, Schaefer and Osman do not explicitly disclose other limitations stated below.

However, in an art of *dynamic classification of sections of software*, Hunt discloses the protocol stack, protocol information for remote communication, the DCOM network protocol which is a superset of Open Group's Distributed Computing Environment Remote Procedure Call (DCE RPC) protocol, and Service Control Manager; All of them are the essential building blocks for distributed Windows applications (e.g., [0076], Lines 21-33; Fig. 4, elements 108/144 – Service Control Manager, 120 – DCOM Network Protocol).

Further, Hunt discloses continuing the RPC call (e.g., [0077], Lines 1-6, 21-33 – the proxy and stub typically include a protocol stack and protocol information for remote communication, for example, the DCOM network protocol, which is a superset of the Open Group's Distributed Computing Environment Remote Procedure Call (DCE RPC) protocol) to locate the privatized virtual component through a service control manager (SCM); redirecting the RPC message call as it is to the requested call for the component call originated from non secured run-time application and for the component call, which do not have redirecting information in the component redirection table (e.g., Fig. 4, elements 108, 144 – Service Control Manager; [0073] – after instantiation, DCOM supports cross-process or cross-machine communication; [0074] – a local service control manager connects to a remote service control manager, which requests creation of the component through the "CoCreateInstance" API).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Hunt into the Blaser-Schaefer-Osman system to further provide other limitations stated above in the Blaser-Schaefer-Osman's system.

The motivation is that it would enhance the Blaser-Schaefer-Osman's system by taking, advancing and/or incorporating Hunt's system which is coupled with a distributed object system such as Microsoft™ Corporation's Distributed

Component Object Model (DCOM™) to further provide system services that support execution of distributed application once suggested by Hunt (e.g., [0005])

51. **As to claim 31** (Original) (incorporating the rejection in claim 24), please refer to claim **8** as set forth accordingly.

Conclusion

52. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Ben C Wang/
Examiner, Art Unit 2192

/Michael J. Yigdall/
Primary Examiner, Art Unit 2192